

TECHNICAL NOTE

No. TN00-

FILEREAD: SOFTWARE MODULE FOR READING SCENARIO MODEL INPUTS AND OBSERVED DATA FROM TEXT FILES

By

T.J. Doherty

February 2000

U.S. Army Research Institute of Environmental Medicine
Natick, MA 01760-5007

DISTRIBUTION STATEMENT A
Approved for Public Release
Distribution Unlimited

20000211 033

DISCLAIMER

The views, opinions, and/or findings contained in this report are those of the author and should not be construed as an official Department of the Army position, policy or decision unless so designated by other official documentation.

Citations of commercial organizations and trade names in this report do not constitute an official Department of the Army endorsement or approval of the products or services of these organizations.

Qualified requestors may obtain copies of this report from Commander, Defense Technical Information Center (DTIC) (formerly DDC), Cameron Station, Alexandria, Virginia 22314.

DTIC AVAILABILITY NOTICE

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE FEB 00		3. REPORT TYPE AND DATES COVERED Technical Note for period 11Jun98-11Aug98
4. TITLE AND SUBTITLE FILEREAD: SOFTWARE MODULE FOR READING SCENARIO MODEL INPUTS AND OBSERVED DATA FROM TEST FILES			5. FUNDING NUMBERS	
6. AUTHOR(S) TAMMY J. DOHERTY, Ph.D.				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U.S. Army Research Institute of Environmental Medicine Natick, MA 01760-5007			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Medical Reserch and Materiel Command Fort Detrick, MD 21702-5012			10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for Public Release; Distribution is Unlimited			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) <p>SCENARIO is a computer simulation predicting thermal and cardiovascular responses to work in the heat. The present versions of SCENARIO enable model input data to be entered only by keyboard or mouse; there are no mechanisms for obtaining data from electronic files. This limits the use of SCENARIO because manual entry of input data for running multiple simulations or for running simulations in which inputs change many times (i.e., simulations of field and laboratory studies) are very tedious and time consuming. This also precludes the use of parameter optimization to simulate the responses of an individual, which requires direct access to observed data. Comparing model predictions (which are saved in ASCII text files) with observed data (saved in other electronic files) during model development and validation is also extremely tedious. The FILEREAD module described in this report attempts to alleviate these problems. Many simulations can be run in a batch mode through the use of a simulation specification file. For a given simulation, FILEREAD reads model input and observed data that are stored in ASCII tables. Any number of tables may be used to hold the simulation data. The study, test and subject identifiers are defined as key fields for querying records so that more than one study, test, and/or subject may be included in the same file. FILEREAD merges data from more than one file based on date and time fields, which must be sequential within each file for a given study/test/subject combination. Variables may reside in any order in any of the data tables; extraneous variables in tables are ignored. Variables may be specified in a wide variety of units. Each time FILEREAD is called from SCENARIO, model input data describing a single scene or set of input conditions is passed from FILEREAD to SCENARIO.</p> <p>This Technical note serves the purpose of documenting the FILEREAD algorithm and software module.</p>				
14. SUBJECT TERMS Software, Algorithm, File I/O			15. NUMBER OF PAGES 48	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT	

CONTENTS

EXECUTIVE SUMMARY.....	1
INTRODUCTION	
TERMINOLOGY.....	2
PROBLEM STATEMENT.....	2
PURPOSE.....	2
METHODS	
GENERAL APPROACH.....	3
FILE STRUCTURES.....	3
DESCRIPTION OF THE FILEREAD SOFTWARE.....	5
IMPLEMENTATION.....	11
DISCUSSION	
LIMITATIONS.....	12
PLANNED IMPROVEMENTS.....	12
REFERENCES.....	13
APPENDICES	
A. SCENARIO-Recognized Variable Names.....	A1-2
B. Example Input Data Files.....	B1
C. FILEREAD.BAS Program Listing.....	C1-14

EXECUTIVE SUMMARY

SCENARIO (1, 2) is a computer simulation predicting thermal and cardiovascular responses to work in the heat. The present versions of SCENARIO enable model input data to be entered only by keyboard or mouse; there are no mechanisms for obtaining data from electronic files. This limits the use of SCENARIO because manual entry of input data for running multiple simulations, or for running simulations in which inputs change many times (such as in simulations of field studies and laboratory experiments) are very tedious and time consuming. This also precludes the use of parameter optimization techniques to customize the model to simulate the responses of an individual, which would require direct access to observed data. Comparing model predictions (which are saved in ASCII text files) with observed data (saved in other electronic files) during model development and validation is also extremely tedious.

The FILEREAD module described in this report attempts to alleviate these problems. Through the use of a simulation specification file, many simulations can be run in a batch mode. For a given simulation, FILEREAD reads model input and observed data that are stored in ASCII tables. Any number of tables may be used to hold the simulation data. The study, test, and subject identifiers are defined as key fields for querying records so that more than one study, test, and/or subject may be included in the same file. FILEREAD merges data from more than one file based on date and time fields, which must be sequential within each file for a given study/test/subject combination. Variables may reside in any order in any of the data tables; extraneous variables in tables are ignored. Variables may be specified in a wide variety of units. Each time FILEREAD is called from SCENARIO, model input data describing a single scene or set of input conditions is passed from FILEREAD to SCENARIO. This Technical note serves the purpose of documenting the FILEREAD algorithm and software module.

INTRODUCTION

TERMINOLOGY

The FILEREAD software is intended to be used with current versions of the SCENARIO model (1, 2). SCENARIO predicts the thermal and cardiovascular responses for a given set of subject characteristics and clothing levels to changes in environmental conditions and levels of physical activity. A **scene** is defined by a single set of input conditions. A **simulation** is a sequence of one or more scenes. A **session** is a sequence of one or more simulations.

PROBLEM STATEMENT

The SCENARIO model currently requires input data to be entered by keyboard; there is no mechanism available to enter data from electronic files. This precludes modeling a large number of scenarios or scenarios with large numbers of "scenes" or sequential changes in input conditions. It also precludes development of adaptive models, in which model parameter values are optimized based on errors between predicted and observed data. Even simple comparisons between predicted and observed data are hindered because the user is required to first merge the ASCII file containing predicted values with electronic files containing the observed data using external software packages.

The development of software to enable SCENARIO to read directly from experimental data files is complicated by the variety of ways in which investigators collect and store data in electronic format. Data are often collected at different sampling frequencies. Missing value codes, units of measure, and variables of measure (e.g., relative humidity and dew point temperature as measures of ambient vapor) may differ from one data collection apparatus to another or due to investigator preferences. Finally, from the model's point of view there is usually extraneous information contained in the data files. Neither extensive reformatting of existing data files nor restriction of data file formats (for data to be collected in the future) are practical solutions to this problem.

PURPOSE

The purpose of the FILEREAD program module is to provide the capability to acquire model inputs and observed data from flexibly formatted text files. From the onset, it is understood that it is unrealistic to require extensive reformatting of existing data files or to restrict file formats for data collected in the future. Therefore, this software must:

- Read from one or a multiple of data files for each "scene" (each distinct set of input conditions constitutes one "scene").
- Automatically put scenes together into simulations.
- Enable the user to batch several simulation runs together into a single session.
- Allow for a variety of different units and measures for each variable
- Ignore irrelevant data fields
- Ignore irrelevant records (e.g., wrong subject, study, or experiment).

METHODS

GENERAL APPROACH

The general approach centers around a simulation specification file which defines all the simulations that make up the session. Simulation definitions include the Study, Test, and Subject IDs for querying data records, the number of input data files, the names of the input data files, the name of the output file (for model predictions), the missing value code, and optional input data values. FILEREAD reads one record of the simulation specification file at the start of each simulation. FILEREAD reads one unique set of data, representing one scene, each time it is called from SCENARIO. The end of the simulation occurs when there are no more valid records in the data files. The end of the session occurs when there are no more records in the simulation specification file.

FILE STRUCTURES

Two types of files are used in a session. These are the Simulation Specification File and the input data files.

The Simulation Specification File

All simulation sessions must be defined within a *Simulation Specification File*. FILEREAD prompts the user for the name of this file at the start of each session. The simulation specification file is a comma-delimited text file. The first row contains the column headers. Each subsequent row (record) holds the information for one simulation. The data columns that must be included in the simulation specification file are listed below, in order:

- Study ID
- Test ID
- Subject ID
- Number of experimental data files (one per column)
- Names of the experimental data files
- Name of the model output (prediction) file
- Missing value code

Following the missing value code, one or more optional data columns may be included.

The easiest way to create the simulation specification file is to use Excel or another spreadsheet program to generate a table, then to save this as a comma-delimited text (.csv) file. An example of a simulation specification was generated for the following conditions:

- One simulation is run for each study/test/subject combination.
- There are three input files: one holding information about the weather (weather.csv), one file holding information about the subjects (subjspec.csv), and one holding physiological data for each subject/test combination (e.g., S1E1.csv, S2E2.csv, S1E2.csv, and S2E2.csv, respectively).
- The output is also named using subject/test the combination (e.g., S1E1out.csv, S2E2out.csv, S1E2out.csv, and S2E2out.csv, respectively).

- The missing value code is a blank cell or empty string.
- There are no input data values included in this file.

The resulting Excel file is shown below:

StudyID	TestID	SubjectID	NumInputFiles	InputFile1	InputFile2	InputFile3	OutputFile	Missing ValueCode
Mystudy	E1	s1	3	d:\simdata \s1E1.csv	d:\simdata\ weather.csv	d:\simdata\ \subspec. csv	d:\simdata\ \S1E1out. csv	
Mystudy	E1	s2	3	d:\simdata \s2E1.csv	d:\simdata\ weather.csv	d:\simdata\ \subspec. csv	d:\simdata\ \S2E1out. csv	
Mystudy	E2	s1	3	d:\simdata \s1E2.csv	d:\simdata\ weather.csv	d:\simdata\ \subspec. csv	d:\simdata\ \S1E2out. csv	
Mystudy	E2	s2	3	d:\simdata \s2E2.csv	d:\simdata\ weather.csv	d:\simdata\ \subspec. csv	d:\simdata\ \S2E2out. csv	

Note that once this file is saved to a .csv format, each row in the table will be represented as a row of comma-delimited values. An example of the same simulation in which two of the model input variables, body weight in kg (BW-kg) and height in cm (HT-cm), are included in the simulation specification file is shown below:

StudyID	TestID	SubjectID	Missing ValueCode	BW-kg	HT-cm
Mystudy	E1	s1		75.6	180.3
Mystudy	E1	s2		75.6	180.3
Mystudy	E2	s1		62.3	175.2
Mystudy	E2	s2		62.3	175.2

Experimental Data Files

The format for the experimental data files is flexible. Formatting requirements are listed below:

- The files must be comma-delimited.
- The first row of the data file must contain the variable names.
- Variable names and units designations must conform exactly to the specifications defined in Appendix A.
- Date/Time must be specified using separate Year, Month, Day, Hour, Minute, Second columns (whichever are applicable).
- Data may be located in a single file or in multiple files.
- Data files may contain extraneous columns - any variable names that FILERREAD cannot recognize are ignored.
- Variables need not be in a particular order or located in a particular file. However, if the same variable is located in more than one file, only the last instance of the variable (file order as specified in the Simulation Specification File) is used.

After the first row containing the variable names, each subsequent row contains a data record describing the conditions at one time point. If the file contains data from more than one Study, Test, and/or Subject, then each row should contain the Study ID, the

Test ID, the Subject ID, respectively. Date and Time information should be entered on each row. The dates and times in different data files need not correspond exactly (i.e., there is no need to generate empty records). For example, one file can hold data collected at 1-minute intervals, one file can hold data collected at hourly intervals, and one file can hold information collected once per day (e.g., subject height, weight, etc). Sampling intervals need not be constant within a given data file. However, for purposes of modeling and other analyses, date and time information generally do need to be sequential within a given file. FILEREAD reads the records in the order in which they appear. Samples of input data files corresponding to the example Simulation Specification File in the previous subsection are provided in Appendix B.

DESCRIPTION OF THE FILEREAD SOFTWARE:

FILEREAD is implemented as a Microsoft Quick BASIC program module named FILEREAD.BAS. The GetFileInput function within FILEREAD is called by SCENARIO, which resides in another Quick BASIC module. GetFileInput is the only external function within FILEREAD (i.e., called from another program or module). All other functions within FILEREAD.BAS are internal (i.e., called directly or indirectly by GetFileInput).

An example of how GetFileInput is called from within the SCENARIO module is shown below:

```
'Ask whether the user wants to enter data via keyboard or ASCII file:
CALL KeyboardOrFileInput(FileOrKeybd$)

WHILE DoQuit = FALSE
  IF FirstScene = TRUE THEN
    DoneSim = FALSE
    CALL ClearOutputs
    GOSUB Parameters      'Read in physical and physiological constants
    GOSUB Defaults        'Read default inputs
    GOSUB InitialZero     'Read default initial conditions
  END IF
  'get 1 scene worth of input data
  SELECT CASE FileOrKeybd$
    CASE "k"
      CALL GetKeyBoardInput
    CASE "f"
      CALL GetFileInput
  END SELECT
  IF DoQuit = FALSE AND DoneSim = FALSE THEN
    GOSUB RunScenario
  END IF
  IF (DoQuit = FALSE AND DoneSim = TRUE) OR LastScene = TRUE THEN
    CLOSE #OutputFileNum%
    FirstScene = TRUE
  END IF
WEND
```

In this section of code, the user is asked whether they would like to enter input data via keyboard or ASCII file. The answer is returned as the string variable FileOrKeybd\$, which contains the value "k" for keyboard, or "f" for file. DoQuit is set to FALSE at the start of the SCENARIO program, while FirstSim and FirstScene are set to TRUE. The While Loop structure keeps the program running through new scenes and simulations until the user decides to quit (keyboard entry mode), or the last simulation in the ASCII simulation specification file has been run. The statements under the first IF statement

(IF FirstScene=TRUE) set SCENARIO up for a new simulation. The statements under the SELECT CASE statement get the model inputs for running one scene. GetFileInput is called when FileOrKeybd\$ is equal to "f." The remainder of statements within the While Loop run the Scenario model for one scene and set up SCENARIO for a new simulation when the current simulation is completed.

The following subsections describe the functions and subroutines in FILEREAD.BAS. The complete program listing of FILEREAD.BAS is provided in Appendix C. The GetFileInput subroutine is described first, with other functions and subroutines described in the order in which they are called.

GetFileInput

A flow chart representation of GetFileInput is shown in Figure 1. Each block with double side bars represents a function which is described in detail in subsequent subsections. FirstSim and FirstScene are inputs that are passed to GetFileInput from SCENARIO. FirstSim is set to TRUE if it is the first simulation within the session. FirstScene is set to TRUE if it is the first scene within a simulation.

GetFileInput also calls ReadSimSpecFile which prompts the user for the name of the simulation specification file, opens the file and reads the header information when FirstSim is TRUE.

Whenever FirstScene is TRUE (whether or not FirstSim is TRUE), GetFileInput calls ReadSimSpecFile which reads a record from the simulation specification file. If on trying to read a record from the simulation specification file it is discovered that the end of the file has been reached, DoneSession is set to TRUE and control returns to the calling program, SCENARIO. After successfully reading a record from the Simulation Specification File, GetFileInput will have obtained values for the Study ID, Test ID, and Subject ID, the number of and names of the input data files, the name of the output data file, the missing value code, and any optional data input values. GetFileInput then calls GetInputVarInfo, which determines which of the column headings in the Simulation Specification File correspond to SCENARIO-recognized variable names. Records are read from each of the input data files using the procedure ReadFirstRecords until the current record in each file corresponds to the record at, or immediately preceding, the StartTime (the first time for which all input data files contain data). If it is not FirstScene, then GetFileInput calls ReadAnotherRecord which reads the data file records corresponding to the next time point. Both ReadAnotherRecord and ReadFirstRecords return a string array containing all the information in the current record, info\$(), the current date/time for each file CurrDate(), and the next date/time for each file NextDate().

Once the information for a single time point is placed into info\$(), GetFileInput calls ConvertDataValues, which takes the information from info\$ and converts it into SCENARIO-recognized variables. GetFileInput then calls CalculateEndMin, which takes the information in CurrDate() and NextDate() and computes the length of the current scene in minutes. When there are no more data records, EndMin is set to zero and DoneSim is set to TRUE.

If DoneSim is TRUE, then FirstSim is set to FALSE, CurrentDate is set to zero, and all the input data files are closed. If DoneSim is not TRUE, then the model input

information is posted to the display screen. Program control is then returned to SCENARIO.

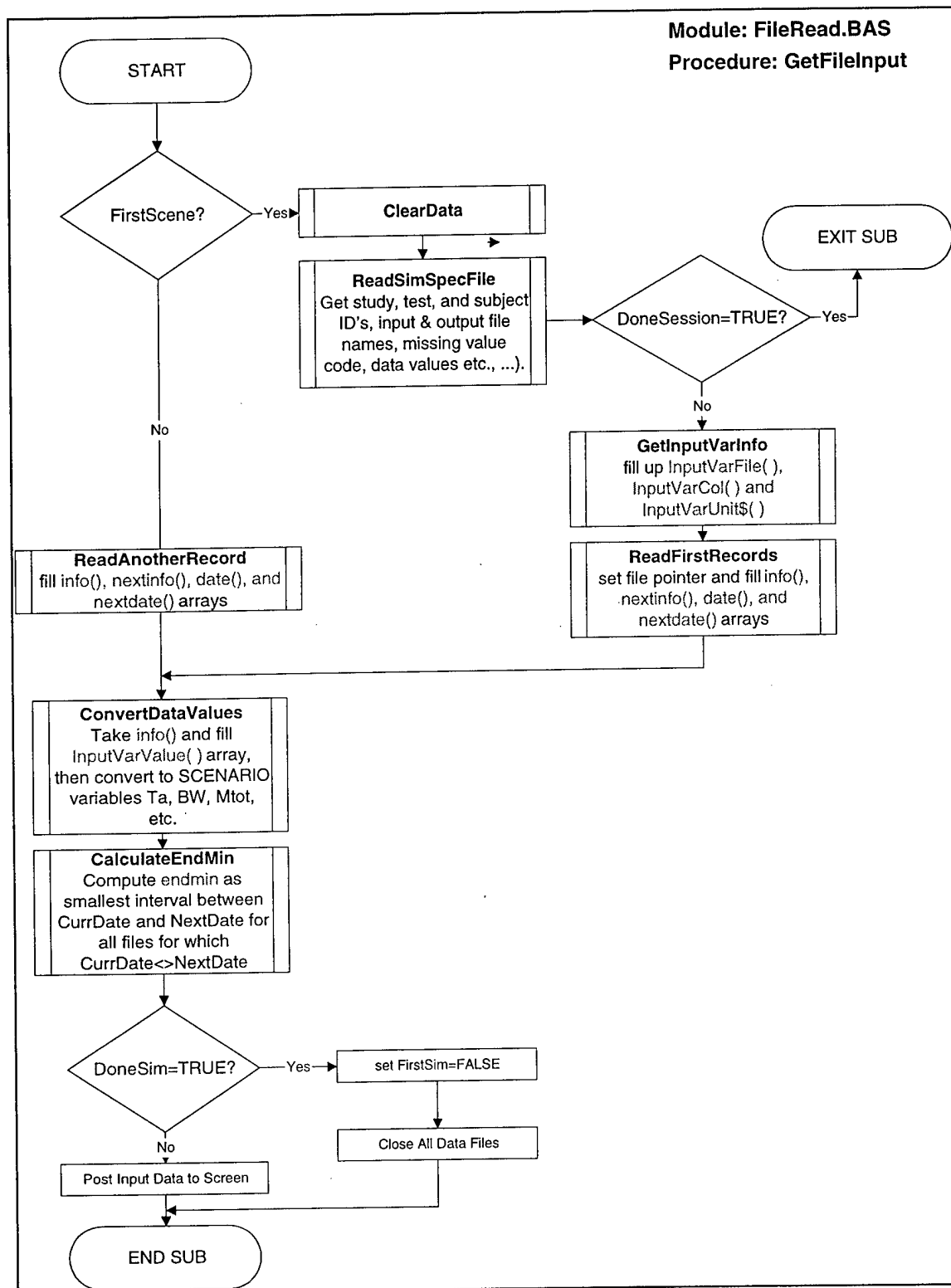


Figure 1. Flow Chart representation of the GetFileInput procedure.

ClearData

ClearData sets all values of info\$() and nextinfo\$() to the empty string. ClearData then sets all values except those corresponding to file number 1 (the Simulation Specification File of InputVarFile and InputVarCol to zero and of InputVarUnit\$() to the empty string.

GetRecognizedVars keeps track of the variables that are currently recognized by SCENARIO. The total number of recognized variables (currently 37) is assigned to NumRecognizedVars%. GetRecognizedVars dimensions the InputVarName\$, InputVarFile, InputVarCol, InputVarUnit\$, and InputVarValue arrays to NumRecognizedVars%. The names of the SCENARIO-recognized variables are assigned to InputVarName\$. The Index for each of the variables is assigned to IX.*, where * is replaced with the SCENARIO variable name. For example, if InputVarName\$(4) is assigned "Vair", then IX.Vair is assigned the value of 4.

ReadSimSpecFile

The first time ReadSimSpecFile is called (i.e., when FirstSim is TRUE), the simulation specification file is opened as file number 1. ReadSimSpecFile calls GetInputVarInfo, which reads the header line into the variable varnames\$. varnames\$ is passed to the procedure ParseVarNames, which parses the varnames\$ string into individual, comma-delimited column headings. ParseVarNames examines each column heading and determines whether it corresponds to one of the SCENARIO-recognized variable names. If so, ParseVarNames assigns appropriate values for InputVarFile, InputVarCol, and InputVarUnit\$. NumVarNames(1) is assigned to the number of variables found in the Simulation Specification File.

Whether or not FirstSim is TRUE, ReadSimSpecFile reads the information in one record of the Simulation Specification File. If the end of file marker is reached, then DoneSession is set to TRUE, and program control returns to the calling procedure. If the end of file marker is not reached, then the StudyID, TestID, SubjectID, Number of Input Files, Input File Names, Output File Name, Missing Value Code, as well as any input data values in the simulation specification file are returned to the calling procedure.

GetInputVarInfo

On the first pass through the procedure, GetInputVarInfo calls GetRecognizedVars, which assigns the current SCENARIO-recognized variable names to InputVarName\$(), and the indices for SCENARIO-recognized variables to IX.*, where * is the variable name (for example, the array index corresponding to the variable Ta is IX.Ta). On each call to the procedure (whether or not it is the first pass through) GetInputVarInfo opens the specified input data file (unless the FileNum corresponds to the Simulation Specification File, which is already open). GetInputVarInfo then reads the entire header line into the string varnames\$. While the length of varnames\$ is greater than zero, GetInputVarInfo assigns the leftmost portion of the string, up to the next comma, to varname\$. If there is no comma, then the entire varnames\$ string is assigned to varname\$ and LastName is set to TRUE. The portion of varnames\$ that

was assigned to varname\$ is removed (i.e., varnames\$ is reduced by one variable name after each call to GetNextVarName). If varname\$ matches one of the recognized date/time or query variables, GetInputVarInfo assigns appropriate values to StudyIDCol, SubjectIDCol, TestIDCol, YearCol, MonthCol, DayCol, HourCol, MinuteCol, or SecondCol, as appropriate. For all other values of varname\$, GetVarInfo parses the units information from varname\$ by calling GetUnits. indexx% is then set to the array index for the corresponding SCENARIO-recognized variable. If an indexx is not equal to zero (i.e., varname\$ is recognized by SCENARIO), then InputVarFile(indexx%) is set to the current file number, InputVarCol(indexx%) is set to the current column number, and InputVarUnit\$(indexx%) is set to the units information parsed from varname\$. As an example, the following array assignments would result from examining the header information from file number 3, containing the column headings "Minute", "Vair-m/s" and "Tdb-F" in columns 1, 3 and 6:

InputVarName\$:

tdb	tmr	tg	vair	rh	pvap	...
-----	-----	----	------	----	------	-----

InputVarFile:

3	0	0	3	0	0	...
---	---	---	---	---	---	-----

InputVarCol:

3	0	0	6	0	0	...
---	---	---	---	---	---	-----

InputVarUnit\$:

F			m/s			...
---	--	--	-----	--	--	-----

ReadFirstDataRecords

The purpose of ReadFirstDataRecords is to establish the common start time (i.e., time at which all the files have some data) and to set the current record for each file at the start time. ReadFirstDataRecords first reads one record from each of the data input files using ReadNextRecord. StartDate is set to the largest CurrDate value. Another record is read from each of the data input files to provide information on the next time/date value (NextDate). Records are then read from each data input file, as needed, until NextDate>StartDate. Whenever a new record is read, current values of nextinfo\$ and NextDate are first assigned to info\$ and CurrDate, respectively. After this process is completed, the information in info\$ () and CurrDate () corresponds to a time point at, or just prior to, StartDate, and nextinfo\$ and NextDate correspond to the first records after StartDate.

ReadNextRecord

ReadNextRecord is called from within ReadFirstDataRecords and also from ReadAnotherRecord (described below). FileNum%, NumVarNames, StudyID, TestID, and SubjectID are passed to ReadNextRecord from the calling procedure. ReadNextRecord first assigns all current values of info\$ () to lastinfo\$ (). Next, if the end of file marker has not been reached, then the next record is read into info\$ (). Otherwise, endfile(FileNum) is set to TRUE. If a StudyID, TestID or SubjectID was specified in the simulation specification file and the data file contains one or more of these fields, then the record that was read must contain appropriate values in these fields. If one or more fields do not contain appropriate values, then another record will

be read. This process will continue until a valid record is read or the end of file marker is reached. If the end of file is reached without obtaining a valid info\$ () string, then info\$ () is reassigned its original values that were saved in lastinfo\$ (). Otherwise, CurrDate is obtained using the Datee function and control is returned to the calling procedure.

ReadAnotherRecord

For any input data files for which NextDate() is equal to the EndDate for the scene, ReadAnotherRecord assigns values of nextinfo\$ and NextDate to info\$ and CurrDate, respectively then reads calls ReadNextRecord to obtain new values for nextinfo\$ () and NextDate().

ConvertDataValues

ConvertDataValues takes info\$() and InputMissingValueCode\$ as inputs and produces values for all SCENARIO input variables. For each SCENARIO-recognized variable, i, FileNum is set to InputVarFile(i), and Col is set to InputVarCol(i). If both of these values are non-zero, then InputVarValue is set to the value contained in info\$(FileNum,Col). If info\$(FileNum,Col) equals the InputMissingValueCode\$ and the variable is one of the recognized "input" variables (i.e., Tdb, ...), then InputVarValue is not assigned any value (i.e., the previous value remains). If info\$(FileNum,Col) equals the InputMissingValueCode\$ but the variable is an "observed" variable (i.e., Tcr, Tsk, ...), then InputVarValue is set to the SCENARIOMissingValueCode. Special cases are used for variables that are entered as strings (i.e., Gender and Phase). Once all values have been assigned, CalculateWeather, CalculateSubjectInfo, and CalculateMetab are called to convert data entered by the user (i.e., user-entered variables and units) into variables and units appropriate for model inputs to scenario.

CalculateEndMin

CalculateEndMin calculates EndMin, which is the time interval between the current record and the next record. If there are more than one data input files, then EndMin is set to the smallest interval. In SCENARIO, EndMin is the length (in time) of the current scene. Note that when all files are at the end of file mark, EndMin will equal zero and DoneSim will be set to TRUE.

IMPLEMENTATION

The 1995 version of SCENARIO (2) that was originally written in Borland's Turbo BASIC was converted to Microsoft Quick BASIC and renamed SCENDOS.BAS. The keyboard data entry routines were teased out of SCENDOS.BAS and placed in a separate program module named KEYBRD.BAS. The work described in this report resulted in FILEREAD.BAS being added as a third Quick BASIC program module. The make file designating these three modules as components of the SCENDOS program and designating SCENDOS.BAS as the main module is named SCENDOS.MAK. These files are saved as plain text and may be viewed by any text viewer or word processing software. Microsoft Quick BASIC is included (as of this date) with all Microsoft DOS and Windows products, allowing the source code to be modified and recompiled as necessary. The SCENDOS source code as well as the compiled

program, SCENDOS.EXE, are available on the USARIEM shared drive in the subdirectory SCENDOS.99.

DISCUSSION

LIMITATIONS

To date, the FILEREAD module has been used with SCENARIO to simulate real experiments and to perform sensitivity analyses. Several problems have been noted:

1. It is inconvenient to format date and time columns into separate year, month, day, hour, minute, and second columns.
2. It is inconvenient to rename variables using the standard set of file names and unit designations in Appendix A.
3. When exporting data from Microsoft Excel (Microsoft Office97) to a comma-delimited text file, placeholders may or may not be created for missing data (blank cells) at the end of the record. FILEREAD expects the empty placeholders and misreads data in subsequent records when they are absent.

PLANNED IMPROVEMENTS

To address the problems noted above, the following enhancements are planned for the next revision:

1. In addition to entering times and dates as separate year, month, day, hour, minute, and second columns, add the option of entering times and dates using any of the formats recognized by Microsoft Excel (i.e., as time or date strings, or as date numbers).
2. Utilize a variable mapping file (VMF) to correlate variable names as entered in the input data files, with names that SCENARIO recognizes. Units of measure could also be specified in the VMF. One VMF would be specified for each input data file, although many data files could utilize the same VMF.
3. Modify the program to accommodate missing place holders or read data directly from spreadsheet applications.
4. Create a user-friendly application for creating the Simulation Specification File.

REFERENCES

1. Kraning K.K. A computer simulation for predicting the time course of thermal and cardiovascular responses to various combinations of heat stress, clothing, and exercise. USARIEM Technical Report T13-91, 1991.
2. Kranning K.K. Validation of mathematical models for predicting physiological events during work and heat stress. USARIEM Technical Report T95-18, 1995.

APPENDIX A

SCENARIO-Recognized Variable Names

The list of SCENARIO-recognized variable names is provided below. In the input data files, the standard variable name should be followed by a dash (hyphen) and the standard units string (where appropriate). For example, dry bulb temperature in °C would be expressed as Tdb-C. If the unit is omitted, the default unit is assumed. Variable and unit names are not case sensitive.

Key Fields

name	Description	potential units
StudyID	Study identifier string	
TestID	Test identifier string	
SubjectID	Subject identifier string	
year	Year	
month	Month	
day	Day	
hour	Hour	
minute	Minute	
second	Second	

Weather Data

name	Description	potential units
Tdb	dry bulb temperature	F,C
Tmr	mean radiant temperature	F,C
Tg	black globe temperature	F,C
Twb	psychometric wet bulb temperature	F,C
Tnwb	natural wet bulb temperature	F,C
Tdp	dew point temperature	F,C
RH	percent relative humidity	
Pvap	saturated vapor pressure	mmHg, Pa, dynes/cm2, atm
Vair	air movement velocity	knots, mph, m/s, f/s

Note that all data will be read before conversions are done; the column order is not important.

Activity Data

name	Description	potential units
HR	heart rate in beats per minute	
Load	weight of everything worn or carried	lbs, Kg
Grade	% grade of terrain	
Vmove	velocity of movement	knots, mph, m/s, f/s

VO2	oxygen uptake in L/min	
Mtot	total metabolic rate	Met, W, Kcal/h, cal/s
Mrst	resting metabolic rate	Met, W, Kcal/h, cal/s
Mext	external work rate as absolute value or % of Mtot	Met, W, Kcal/h, cal/s
PctEff	percent efficiency	
Mwork	metabolic cost of work	Met, W, Kcal/h, cal/s
WorkMode\$	character: r=rest, e=ergometer, a=armwork, t=treadmill, f=free-walking	

Subject Information

name	Description	potential units
gender	gender ("m"=male, "f"=female)	
BW	body weight (nude)	lbs, Kg
HT	body height	in, m, cm
PctFat	% body fat	
Age	Age in years	
Accl	acclimation state (0=none, 1=partial, 2=full)	
VO2max	maximal oxygen uptake	L/min
Iclo	total clothing insulation	clo
Im	clothing moisture permeability index	

Observed Data (Used for Model Validation)

name	Description	potential units
HR	heart rate in beats per minute	
Tpill	core temperature pill	C,F
Tre	rectal temperature	C,F
Tes	esophageal temperature	C,F
Ttym	tympanic temperature	C,F
Tsk	mean skin temperature	C,F

APPENDIX B

Example Input Data Files

WEATHER.CSV

```
station,year,month,day,hour,minute,Tdb-F,RH,Vair-mpH
4,1997,9,22,10,55,73.6,72,5
4,1997,9,22,11,0,73.9,71,6
4,1997,9,22,11,5,73.7,72,7
4,1997,9,22,11,10,73.6,72,6
4,1997,9,22,11,15,73.8,71,4
4,1997,9,22,11,20,74.3,70,5
4,1997,9,22,11,25,74.5,69,6
4,1997,9,22,11,30,74.4,70,6
4,1997,9,22,11,35,74.2,70,6
4,1997,9,22,11,40,74.2,70,6
3,1997,9,22,11,45,74.2,70,0
3,1997,9,22,11,50,74.5,69,0
3,1997,9,22,11,55,74.7,69,2
```

SUBSPEC.CSV

```
OpID,SubjectID,Month,Day,Year,Hour,Minute,Pack,WIN CE,Pill,TotWt-
lbs,Xtal,Age,HT-cm,BW-Kg,PctFat,VO2max-L/m
Mystudy,s3,9,22,1997,6,0,5,1,37291345,213.6,9877,24,180,81.91,20.9,4.1
Mystudy,s1,9,22,1997,6,0,2,7,33241336,194.4,9842,21,175,68.45,15.9,4.03
Mystudy,s4,9,22,1997,6,0,3,3,34381336,197.2,9849,22,173,70,16.6,5.9
Mystudy,s2,9,22,1997,6,0,7,6,36791351,237,9953,21,188,84.82,18.3,4.29
```

DATA FOR 1 SUBJECT (S4DATA.CSV)

```
OpID,SubjectID,month,day,year,hour,minute,latd,latm,lond,lonm,grade,distance,V
move-m/s,elevation,hr,Tpill-C,tc,steps,cloco,Mwork-W,WorkMode
mystudy,s4,9,22,1997,10,50,32,16.655,84,56.868,0,0.13424,0.00224,74,89,37.3252
1333,-1,0,-1,0,r
mystudy,s4,9,22,1997,10,51,32,16.655,84,56.868,0,0.13424,0.00224,74,90,37.2621
6,-1,0,-1,0,r
mystudy,s4,9,22,1997,10,52,32,16.657,84,56.868,0,3.70806,0.0618,74,84,37.28081
667,-1,0,-1,0,r
mystudy,s4,9,22,1997,10,53,32,16.658,84,56.868,0,1.85767,0.03096,74,98,37.2648
4333,170.5,4,-1,0,r
mystudy,s4,9,22,1997,10,54,32,16.659,84,56.869,0,2.42564,0.04043,74,92,37.2876
8333,-1,0,-1,0,r
mystudy,s4,9,22,1997,10,55,32,16.67,84,56.872,0,20.91788,0.34863,74,103,37.179
64333,480.3714,24,-1,0,r
mystudy,s4,9,22,1997,10,56,32,16.711,84,56.879,0,76.76043,1.27934,74,110,37.21
015,688.0364,46,4.87933,340.4977,f
mystudy,s4,9,22,1997,10,57,32,16.766,84,56.876,0,102.02334,1.700s3,74,109,37.1
9408667,637.95,49,5.4009,376.8945,f
mystudy,s4,9,22,1997,10,58,32,16.8,84,56.866,0.0154,64.92828,1.08214,75,103,37
.24557667,694.6833,44,5.14584,359.0956,f
mystudy,s4,9,22,1997,10,59,32,16.817,84,56.864,0,31.65649,0.52761,75,109,37.24
126,689.7501,51,5.04594,352.1244,f
mystudy,s4,9,22,1997,11,0,32,16.832,84,56.866,0,27.97109,0.46618,75,105,37.316
37,691.7333,49,4.841,337.823,f
```

APPENDIX C

FILEREAD.BAS Program Listing

```

DECLARE FUNCTION Study$ (FileNum%, info$())
DECLARE FUNCTION Test$ (FileNum%, info$())
DECLARE FUNCTION HAVE! (FirstScene!, HAVE.X!, IX.X!)
DECLARE SUB CalculateEndMin (NumInputFiles%, CurrentDate!, CurrDate!(), NextDate!(), EndDate!,
endmin!)
DECLARE SUB GetFileName (prompt$, filename$)
DECLARE SUB ConvertDataValues (FirstSim, FirstScene, info$(), InputMissingValueCode$, Ta!, Pvp!,
Vair!, Tmr!, BW!, HT!, SA!, Age!, PctFat!, VO2max!, Accl!, Iclo!, Im!, load!, grade!, Vmove!,
MWork!, Mtot!, Mrst!, Mext!, PctEff!, WorkMode$)
DECLARE SUB GetInputVarValue (FirstScene, info$(), InputMissingValueCode$)
DECLARE SUB ReadFirstRecords (FileNum%, SubjectID$, info$(), CurrDate!(), nextinfo$(),
NextDate!(), CurrentDate!)
DECLARE SUB Post (x%, y%, d1%, d2%, x!)
DECLARE SUB ClrVal (x%, y%)
DECLARE SUB PostInputData (Ta!, Tmr!, Pvp!, Vair!, Iclo!, Im!, BW!, HT!, SA!, Mtot!, Mext!,
Vmove!, grade!, load!, Age!, StMin!, PctFat!, rptmin!, endmin!)
DECLARE SUB GetEndMin ()
DECLARE SUB GetObservedData (FirstSim!, FirstScene!, info$(), InputMissingValueCode$)
DECLARE SUB ClearData (MaxNumFiles%, MaxNumVars%, info$(), nextinfo$(), InputVarFile!(),
InputVarCol!(), InputVarUnit$(), InputVarValue!())
DECLARE SUB GetPvp ()
DECLARE FUNCTION ToMmHg! (indexx!)
DECLARE SUB GetTmr ()
DECLARE SUB GetVair ()
DECLARE FUNCTION ToMps! (indexx!)
DECLARE SUB GetTa ()
DECLARE FUNCTION ToDegC! (indexx!)
DECLARE SUB GetNextVarName (varnames$, varname$, LastName!)
DECLARE SUB GetUnits (varname$, unit$)
DECLARE SUB ReadDefaultVarNames ()
DECLARE SUB ReadAnotherRecord (NumInputFiles%, SubjectID$, nextinfo$(), info$(), NextDate!(),
CurrDate!(), EndDate!, CurrentDate!)
DECLARE SUB ReadNextRecord (FileNum%, NumVarNames!, SubjectID$, info$(), CurrDate!())
DECLARE SUB CalculateSubjectInfo (FirstScene, BW!, HT!, SA!, Age!, PctFat!, VO2max!, Accl!,
Iclo!, Im!)
DECLARE SUB CalculateWeather (FirstScene, Ta!, Pvp!, Vair!, Tmr!)
DECLARE FUNCTION Date$ (FileNum%, info$())
DECLARE FUNCTION Watts! (Index!, x!)
DECLARE SUB GetInputValues (info$(), InputMissingValueCode$, Ta!, Pvp!, Vair!, Tmr!, BW!, HT!,
SA!, Age!, PctFat!, VO2max!, Accl!, Iclo!, Im!, load!, grade!, Vmove!, MWork!, Mtot!, Mrst!,
Mext!, PctEff!, WorkMode$)
DECLARE SUB ReadSimSpecFile (FirstSim!, StudyID$, TestID$, SubjectID$, NumInputFiles%,
InputFile$(), InputMissingValueCode$, info$(), DoneSession!)
DECLARE FUNCTION Subject$ (FileNum%, info$())
DECLARE FUNCTION VarIndex$ (varname$)
DECLARE SUB GetRecognizedVars ()
DECLARE SUB CalculateEndMin (NumInputFiles%, CurrentDate!, CurrDate!(), NextDate!(), EndDate!,
endmin!)
DECLARE SUB CalculateMetab (FirstScene!, BW!, load!, grade!, Vmove!, MWork!, Mtot!, Mrst!, Mext!,
PctEff!, WorkMode$)
DECLARE FUNCTION JULIAN! (Month!, Day!, Year!)
DECLARE SUB GetInputVarInfo (FileNum%)
DECLARE SUB OpenInputFiles (FirstFileNum%, LastFileNum%)
DECLARE SUB SetCurrentEqualNext (FileNum%, NumVarNames!, nextinfo$(), info$(), NextDate!(),
CurrDate!())

'EXTERNAL FUNCTIONS
DECLARE FUNCTION SatVP! (Temp!)
DECLARE FUNCTION TRIM$ (x$)
DECLARE FUNCTION lc$ (stringg$)
DECLARE FUNCTION FtoC! (Temp!)

COMMON SHARED /General/ TRUE, FALSE, SCENARIOMissingValueCode
COMMON SHARED /inputs1/ Ta, Tmr, Tg, Vair, Pvp, BW, HT, SA, PctFat, VO2max, Age, Gender$,
Phase$, Accl, Iclo, Im
COMMON SHARED /inputs2/ load, grade, Vmove, MWork, Mtot, Mrst, Mext, PctEff, WorkMode$, endmin,
rptmin, dt
COMMON SHARED /inputs3/ OutputFileName$, FileOutputYN$, OutputFileNum%
COMMON SHARED /inputvarinfo/ InputVarName$(), InputVarFile(), InputVarCol(), InputVarUnit$(),
InputVarValue()

```

```

COMMON SHARED /observed/ NumObserved%, observed(), ObservedName$(), ObservedUnit$()
COMMON SHARED /RecVars/ NumRecognizedVars%, ix.FirstObserved%
COMMON SHARED /start/ StartYear, StartDate, UsePandolfEqn
COMMON SHARED /TcoreFile/ TcoreFile%
COMMON SHARED /index1/ IX.Ta, IX.Tmr, IX.Tg, IX.Twb, IX.Tnwb, IX.RH, IX.Pvap, IX.Vair
COMMON SHARED /index2/ IX.HR, IX.load, IX.grade, IX.Vmove, IX.VO2, IX.Mtot, IX.Mrst, IX.Mext,
IX.MWork, IX.WorkMode
COMMON SHARED /index3/ IX.BW, IX.HT, IX.PctFat, IX.Age, IX.gender, IX.Phase, IX.Accl, IX.VO2max,
IX.Iclo, IX.Im, IX.endmin, IX.rptmin
COMMON SHARED /index4/ IX.Tpill, IX.Tre, IX.Tes, IX.Ttym, IX.Tsk
'assumes max # input files is 10, max # cols per file is 25
DIM SHARED NumInputFiles%
DIM SHARED CurrDate(10), NextDate(10), EndFile(10)
DIM SHARED StudyIDCol(10), TestIDCol(10), SubjectIDCol(10), YearCol(10), DayCol(10), MonthCol(10)
DIM SHARED HourCol(10), MinuteCol(10), SecondCol(10)
DIM SHARED NumVarNames(10), InputFile$(10)

```

```

SUB CalculateEndMin (NumInputFiles%, CurrentDate, CurrDate(), NextDate(), EndDate, endmin)
  IF InputVarFile(IX.endmin) > 0 THEN 'endmin entered directly
    endmin = InputVarValue(IX.endmin)
    IF endmin = SCENARIOMissingValueCode THEN
      endmin = 60
    END IF
  ELSE
    EndDate = CurrentDate
    'Set EndDate equal to the smallest NextDate which is not equal to CurrDate
    FOR FileNum% = 2 TO NumInputFiles% + 1
      IF NextDate(FileNum%) > CurrDate(FileNum%) THEN 'not at end of this file
        IF EndDate = CurrentDate THEN 'this is the first file that is not at the end
          EndDate = NextDate(FileNum%)
        ELSE
          IF NextDate(FileNum%) < EndDate THEN EndDate = NextDate(FileNum%)
        END IF
      END IF
    NEXT FileNum%
    numdays = EndDate - CurrentDate
    endmin = numdays * 24 * 60
  END IF
  IF InputVarFile(IX.rptmin) > 0 THEN
    'rptmin entered. Otherwise use default value
    rptmin = InputVarValue(IX.rptmin)
    IF rptmin = SCENARIOMissingValueCode THEN
      rptmin = 1
    END IF
  ELSE
    rptmin = endmin
  END IF
END SUB

```

END SUB

```

SUB CalculateMetab (FirstScene, BW, load, grade, Vmove, MWork, Mtot, Mrst, Mext, PctEff,
WorkMode$) STATIC
STATIC have.Mrst, have.WorkMode, have.Vmove, have.PctEff, have.load, have.grade, have.Mtot

```

T = -1

```

  have.Mrst = HAVE(FirstScene, have.Mrst, IX.Mrst)
  have.WorkMode = HAVE(FirstScene, have.WorkMode, IX.WorkMode)
  have.PctEff = HAVE(FirstScene, have.PctEff, IX.PctEff)
  have.load = HAVE(FirstScene, have.load, IX.load)
  have.grade = HAVE(FirstScene, have.grade, IX.grade)
  have.Vmove = HAVE(FirstScene, have.Vmove, IX.Vmove)
  have.Mtot = HAVE(FirstScene, have.Mtot, IX.Mtot)

```

'get whatever data has been entered directly

```

IF have.Mrst THEN
  Mrst = InputVarValue(IX.Mrst)
  IF InputVarUnit$(IX.Mrst) = "met" THEN
    Mrst = 1.5 * BW
  ELSE
    Mrst = Watts(IX.Mrst, Mrst)
  END IF
END IF

```

```

IF have.WorkMode THEN
    WorkMode$ = CHR$(InputVarValue(IX.WorkMode))
END IF

IF have.PctEff THEN
    PctEff = InputVarValue(IX.PctEff)
END IF

IF have.load THEN
    load = InputVarValue(IX.load)
    IF InputVarUnit$(IX.load) = "lbs" THEN load = load / 2.2
END IF

IF have.grade THEN
    grade = InputVarValue(IX.grade)
END IF

IF have.Vmove THEN
    Vmove = InputVarValue(IX.Vmove)
    SELECT CASE InputVarUnit$(IX.Vmove)
        CASE IS = "knots"
            Vmove = Vmove * .5148
        CASE IS = "mph"
            Vmove = Vmove * .44704
        CASE IS = "f/s"
            Vmove = Vmove * .3048
    END SELECT
END IF

IF have.Mtot THEN
    Mtot = Watts(IX.Mtot, InputVarValue(IX.Mtot))
END IF

IF HAVE.MWork THEN
    MWork = Watts(IX.MWork, InputVarValue(IX.MWork))
END IF

IF HAVE.Mext THEN
    Mext = Watts(IX.Mext, InputVarValue(IX.Mext))
END IF

'estimate work mode
IF NOT have.WorkMode THEN
    IF have.Vmove AND Vmove > .4 THEN WorkMode$ = "f"
END IF

'estimate resting metabolic rate
IF NOT have.Mrst THEN 'must estimate
    IF have.Mtot AND HAVE.MWork THEN
        Mrst = Mtot - MWork
    ELSE
        Mrst = 1.5 * BW 'empirical estimate
    END IF
END IF

LOCATE 22, 1

'estimate total metabolic rate
IF NOT have.Mtot THEN
    IF HAVE.MWork THEN
        Mtot = Mrst + MWork
    ELSEIF WorkMode$ = "f" THEN
        Mtot = Mrst + (2! * (BW + load) * (load / BW) ^ 2) + ((BW + load) * (1.5 * (Vmove) ^ 2 +
        (.35 * grade * Vmove)))
    ELSEIF WorkMode$ = "r" THEN
        Mtot = Mrst
    ELSE
        Mtot = Mrst
    END IF
END IF

'estimate Work Rate
IF NOT HAVE.MWork THEN
    MWork = Mtot - Mrst
END IF

```

```

'estimate external Work Rate
IF NOT HAVE.Mext THEN
  IF Mtot > Mrst THEN 'only compute if working
    IF have.PctEff THEN
      Mext = Mtot * PctEff / 100
    ELSEIF WorkMode$ = "f" THEN 'free-walking
      Mext = .098 * grade * (BW + load) * Vmove
    ELSE
      Mext = Mtot * .2          'if all else fails, use a value of 20%
    END IF
  END IF
END IF
END SUB

SUB CalculateSubjectInfo (FirstScene, BW, HT, SA, Age, PctFat, VO2max, Accl, Iclo, Im)
  STATIC have.BW, have.HT, HAVE.SA, have.Age, have.gender, have.Phase, have.PctFat
  STATIC have.VO2max, have.Accl, have.Iclo, have.Im

  IF HAVE(FirstScene, have.BW, IX.BW) THEN
    'BW entered. Otherwise keep default value
    BW = InputVarValue(IX.BW)
    IF InputVarUnit$(IX.BW) = "lbs" THEN BW = BW / 2.2
  END IF

  IF HAVE(FirstScene, have.HT, IX.HT) THEN
    'HT entered. Otherwise use default value
    HT = InputVarValue(IX.HT)
    IF InputVarUnit$(IX.HT) = "in" THEN HT = HT * 2.54
    IF InputVarUnit$(IX.HT) = "m" THEN HT = HT * 100
  END IF

  'calculate DuBois surface area (BW in Kg, HT in cm)
  SA = .202 * BW ^ .425 * (HT / 100) ^ .725

  IF HAVE(FirstScene, have.Age, IX.Age) THEN
    'Age entered. Otherwise use default value
    Age = InputVarValue(IX.Age)
  END IF

  IF HAVE(FirstScene, have.gender, IX.gender) THEN
    'Gender entered. Otherwise use default value (male)
    Gender$ = CHR$(InputVarValue(IX.gender))
  END IF

  IF HAVE(FirstScene, have.Phase, IX.Phase) THEN
    'menstrual cycle phase entered
    Phase$ = CHR$(InputVarValue(IX.Phase))
  END IF

  IF HAVE(FirstScene, have.PctFat, IX.PctFat) THEN
    'PctFat entered. Otherwise use default value
    PctFat = InputVarValue(IX.PctFat)
  END IF

  IF HAVE(FirstScene, have.VO2max, IX.VO2max) THEN
    'VO2max entered. Otherwise use default value
    VO2max = InputVarValue(IX.VO2max)
  END IF

  IF HAVE(FirstScene, have.Accl, IX.Accl) THEN
    'Accl entered. Otherwise use default value
    Accl = InputVarValue(IX.Accl)
  END IF

  IF HAVE(FirstScene, have.Iclo, IX.Iclo) THEN
    'Iclo entered. Otherwise use default value
    Iclo = InputVarValue(IX.Iclo)
  END IF

  IF HAVE(FirstScene, have.Im, IX.Im) THEN
    'Im entered. Otherwise use default value
    Im = InputVarValue(IX.Im)
  END IF
END SUB

```

```
SUB CalculateWeather (FirstScene, Ta, Pvap, Vair, Tmr)
STATIC Have.Ta, Have.Vair, Have.Tmr, Have.Pvap
```

```
IF HAVE(FirstScene, Have.Ta, IX.Ta) THEN
  'Ta entered. Otherwise use default value
  Ta = ToDegC(IX.Ta)
  Tmr = Ta 'reset the default value for Tmr to Ta
  Tg = Ta ' " " " " " " Tg " "
END IF
```

```
IF HAVE(FirstScene, Have.Vair, IX.Vair) THEN
  'Vair entered. Otherwise use default value
  Vair = ToMps(IX.Vair)
  IF Vair < .1 THEN Vair = .1
END IF
```

```
IF HAVE(FirstScene, Have.Tmr, IX.Tmr) THEN
  'mean radiant temperature has been entered
  Tmr = ToDegC(IX.Tmr)
  k = 1.824 * Vair ^ (.5)
  Tg = (Tmr + k * Ta) / (1 + k)
ELSEIF HAVE(FirstScene, Have.Tg, IX.Tg) THEN
  'black globe temperature has been entered
  Tg = ToDegC(IX.Tg)
  k = 1.824 * Vair ^ (.5)
  Tmr = Tg + k * (Tg - Ta)
END IF
```

```
IF HAVE(FirstScene, Have.Pvap, IX.Pvap) THEN
  'Pvap has been entered
  Pvap = ToMmHg(IX.Pvap)
ELSEIF HAVE(FirstScene, Have.RH, IX.RH) THEN
  'RH has been entered
  RH = InputVarValue(IX.RH)
  Pvap = RH * SatVP(Ta) / 100
ELSEIF HAVE(FirstScene, Have.Tdp, IX.Tdp) THEN
  'dewpoint temperature has been entered
  Tdp = ToDegC(IX.Tdp)
  Pvap = SatVP(Tdp)
ELSEIF HAVE(FirstScene, Have.Twb, IX.Twb) THEN
  'psychometric wet bulb temperature has been entered
  Twb = ToDegC(IX.Twb)
  Pvap = SatVP(Twb) - .674825 * (Ta - Twb)
ELSEIF HAVE(FirstScene, Have.Tnwb, IX.Tnwb) THEN
  'psychometric wet bulb temperature has been entered
  Tnwb = ToDegC(IX.Tnwb)
  Twb = Tnwb - .5 + (Vair <= 1) - .13 * (Tg - Ta)
  Pvap = SatVP(Twb) - .674825 * (Ta - Twb)
END IF
```

```
END SUB
```

```
SUB ClearData (MaxNumFiles%, MaxNumVars%, info$(), nextinfo$(), InputVarFile(), InputVarCol(),
InputVarUnit$(), InputVarValue())
FOR FileNum% = 1 TO MaxNumFiles%
  FOR VarNum% = 1 TO MaxNumVars%
    info$(FileNum%, VarNum%) = ""
    nextinfo$(FileNum%, VarNum%) = ""
  NEXT VarNum%
NEXT FileNum%
FOR VarNum% = 1 TO NumRecognizedVars%
  IF InputVarFile(VarNum%) <> 1 THEN
    InputVarFile(VarNum%) = 0
    InputVarCol(VarNum%) = 0
    InputVarUnit$(VarNum%) = ""
  END IF
  InputVarValue(VarNum%) = SCENARIOMissingValueCode
NEXT VarNum%
```

```
END SUB
```

```
SUB ConvertDataValues (FirstSim, FirstScene, info$(), InputMissingValueCode$, Ta, Pvap, Vair,
Tmr, BW, HT, SA, Age, PctFat, VO2max, Accl, Iclo, Im, load, grade, Vmove, MWork, Mtot, Mrst,
Mext, PctEff, WorkMode$)
  'This subroutine first takes data from info$() and places it in the appropriate position
```

```

'in the InputVarValue array using GetInputVarValue.

CALL GetInputVarValue(FirstScene, info$(), InputMissingValueCode$)
'The subroutines CalculateWeather, CalculateMetab, and CalculateSubjectInfo
'take the values from InputVarValue() and pass them to SCENARIO-recognized variable
'names such as Ta, Vair, BW, ...
CALL CalculateWeather(FirstScene, Ta, Pvap, Vair, Tmr)
CALL CalculateSubjectInfo(FirstScene, BW, HT, SA, Age, PctFat, VO2max, Accl, Iclo, Im)
CALL CalculateMetab(FirstScene, BW, load, grade, Vmove, MWork, Mtot, Mrst, Mext, PctEff,
WorkMode$)

CALL GetObservedData(FirstSim, FirstScene, info$(), InputMissingValueCode$)

END SUB

FUNCTION Datee (FileNum%, info$())

IF YearCol(FileNum%) > 0 THEN Year = VAL(info$(FileNum%, YearCol(FileNum%)))
IF DayCol(FileNum%) <> 0 THEN Day = VAL(info$(FileNum%, DayCol(FileNum%)))
IF MonthCol(FileNum%) <> 0 THEN Month = VAL(info$(FileNum%, MonthCol(FileNum%)))
IF HourCol(FileNum%) <> 0 THEN Hour = VAL(info$(FileNum%, HourCol(FileNum%)))
IF MinuteCol(FileNum%) <> 0 THEN Minute = VAL(info$(FileNum%, MinuteCol(FileNum%)))
IF StartYear = 0 THEN StartYear = Year
Timee = Hour + Minute / 60 + Second / 3600 'hours elapsed since midnight
ThisDate = JULIAN(Month, Day, Year) + Timee / 24 'Days elapsed since start of current year
IF Year > StartYear THEN
    FOR y = StartYear TO Year - 1
        ThisDate = ThisDate + JULIAN(12, 31, y) 'Date is Days elapsed since StartYear
    NEXT y
END IF
Datee = ThisDate
END FUNCTION

SUB GetFileInput (FirstSim, FirstScene, LastScene, DoneSim, DoneSession) STATIC
'-----
'GetFileInput is the main function for getting input data from ASCII data files
',
',
'Inputs:
' FirstSim - integer which takes on the value of TRUE if this is the first simulation
' FirstScene - integer which takes on the value of TRUE if this is the first scene within a
simulation.
',
'Outputs:
' LastScene - integer which takes on the value of TRUE if this is the last scene within a
simulation.
' DoneSim - integer which takes on the value of TRUE if this is there are no more data to return
for the current simulation.
' DoneSession - integer which takes on the value of TRUE if this is there are no more data to
return for the current session.
' Environmental Input Variables:
' Ta - Dry bulb temperature in °C
' Tmr - Mean radiant temperature in °C
' Tg - Black globe temperature in °C
' Vair - Air speed in m/s
' Pvap - Water vapor pressure in mm Hg
' Subject And Clothing Characteristics:
' BW - Body weight in kg
' HT - Body height in cm
' SA - Body surface area in m2
' PctFat - Percent body fat
' VO2max - Maximal Oxygen Uptake
' { Age - Age
' Gender$ - Gender ("m" for male, "f" for female)
' Phase$ - Menstrual cycle phase
' Accl - Acclimation state (0 for unacclimated, 1 for partial, 2 for fully acclimated)
' Iclo - Clothing insulation in clo units
' Im - Moisture permeability coefficient Im/clo
' Activity Parameters:
' load - load in kg
' grade - terrain grade as a percent
' Vmove - ground speed in m/s
' MWork - Metabolic cost of work in Watts
' Mtot - Total metabolic rate in Watts
' Mrst - Resting metabolic rate in Watts
' Mext - The amount of external work done in Watts
' PctEff - The percent work efficiency

```

```

' WorkMode$ - Work mode ("r" for rest, "w" for walking)
' Simulation Parameters:
' endmin - length of the scene in minutes
' rptmin - the interval to display results and write to data files in minutes
' dt - the iteration interval in minutes
' OutputFileName$ - Name of the output file
' FileOutputYN$ - Flag to determine whether results should be written to an output file
' OutputFileNum% - File number corresponding to the output file for the purpose of OPEN
statements
'-----
DIM info$(10, 25), nextinfo$(10, 25)
IF FirstScene = TRUE THEN
    CALL ClearData(10, 25, info$(), nextinfo$(), InputVarFile(), InputVarCol(), InputVarUnit$(),
InputVarValue())
    CALL ReadSimSpecFile(FirstSim, StudyID$, TestID$, SubjectID$, NumInputFiles%, InputFile$(),
InputMissingValueCode$, info$(), DoneSession)
    IF DoneSession = TRUE THEN
        EXIT SUB
    ELSE
        FOR FileNum% = 2 TO NumInputFiles% + 1
            CALL GetInputVarInfo(FileNum%)
        NEXT FileNum%
        CALL ReadFirstRecords(NumInputFiles%, SubjectID$, info$(), CurrDate(), nextinfo$(),
NextDate(), CurrentDate)
    END IF
    ELSE 'not FirstScene
        'for files whos next dates are equal to EndDate (=/- dt/2), read another record
        CALL ReadAnotherRecord(NumInputFiles%, SubjectID$, nextinfo$(), info$(), NextDate(),
CurrDate(), EndDate, CurrentDate)
    END IF
    'Do the following for all scenes
    CALL ConvertDataValues(FirstSim, FirstScene, info$(), InputMissingValueCode$, Ta, Pvpap, Vair,
Tmr, BW, HT, SA, Age, PctFat, VO2max, Accl, Iclo, Im, load, grade, Vmove, MWork, Mtot, Mrst,
Mext, PctEff, WorkMode$)
    CALL CalculateEndMin(NumInputFiles%, CurrentDate, CurrDate(), NextDate(), EndDate, endmin)
    IF endmin = 0 AND NumInputFiles% > 0 THEN DoneSim = TRUE
    IF DoneSim = TRUE THEN
        FOR FileNum% = 2 TO NumInputFiles% + 1
            CLOSE #FileNum%
        NEXT FileNum%
    ELSE
        CALL PostInputData(Ta, Tmr, Pvpap, Vair, Iclo, Im, BW, HT, SA, Mtot, Mext, Vmove, grade,
load, Age, StMin, PctFat, rptmin, endmin)
    END IF

    IF NumInputFiles% = 0 THEN LastScene = TRUE
END SUB

SUB GetFileName (prompt$, filename$)
LOCATE 22, 4
PRINT "Enter " + prompt$ + ", including drive and path names "
PRINT "(example: d:\thispath\thisfile.csv).";
INPUT filename$
END SUB

SUB GetInputVarInfo (FileNum%)
STATIC BeenHereBefore

IF BeenHereBefore = FALSE THEN
    BeenHereBefore = TRUE
    GetRecognizedVars
END IF

'open the input file if FileNum%>1
IF FileNum% > 1 THEN
    OPEN InputFile$(FileNum% - 1) FOR INPUT AS #FileNum%
    EndFile(FileNum%) = FALSE
END IF

LINE INPUT #FileNum%, varnames$
IF FileNum% = 1 THEN 'special case for simulation specification file
    z% = INSTR(varnames$, "MissingValueCode")
    IF LEN(varnames$) > z% + 16 THEN
        varnames$ = TRIM$(RIGHT$(varnames$, LEN(varnames$) - z% - 16))
    ELSE

```

```

        varnames$ = ""
    END IF
END IF

LastName = -(LEN(varnames$) = 0) 'will be equal to TRUE if varnames$ is empty
I% = 0
WHILE LastName = FALSE
    I% = I% + 1
    CALL GetNextVarName(varnames$, varname$, LastName)
    SELECT CASE varname$
        CASE IS = "studyid"
            StudyIDCol(FileNum%) = I%
        CASE IS = "testid"
            TestIDCol(FileNum%) = I%
        CASE IS = "subjectid"
            SubjectIDCol(FileNum%) = I%
        CASE IS = "year"
            YearCol(FileNum%) = I%
        CASE IS = "month"
            MonthCol(FileNum%) = I%
        CASE IS = "day"
            DayCol(FileNum%) = I%
        CASE IS = "hour"
            HourCol(FileNum%) = I%
        CASE IS = "minute"
            MinuteCol(FileNum%) = I%
        CASE IS = "second"
            SecondCol(FileNum%) = I%
        CASE ELSE 'not StudyID, TestID, SubjectID, or date/time
            CALL GetUnits(varname$, unit$)
            indexx% = VarIndex$(varname$) 'checks to see if varname$ is one of the
                                         'recognized names and, if so, gets the

            IF indexx% <> 0 THEN
                InputVarFile(indexx%) = FileNum%
                InputVarCol(indexx%) = I%
                InputVarUnit$(indexx%) = unit$
            END IF
    END SELECT
WEND
NumVarNames(FileNum%) = I%

END SUB

SUB GetInputVarValue (FirstScene, info$(), InputMissingValueCode$)
    'This subroutine takes information from info$() and puts it into InputVarValue()

    FOR z% = 1 TO NumRecognizedVars%
        FileNum% = InputVarFile(z%)
        Col% = InputVarCol(z%)
        IF FileNum% > 0 AND Col% > 0 THEN
            IF z% = IX.WorkMode OR z% = IX.gender OR z% = IX.Phase THEN 'char input
                IF lc$(info$(FileNum%, Col%)) <> "" THEN
                    InputVarValue(z%) = ASC(lc$(info$(FileNum%, Col%)))
                END IF
            ELSEIF z% >= ix.FirstObserved% THEN 'observed variables - carry over missing values
                IF (TRIM$(info$(FileNum%, Col%)) <> InputMissingValueCode$) AND
                    (LEN(TRIM$(info$(FileNum%, Col%))) > 0) THEN
                    InputVarValue(z%) = VAL(info$(FileNum%, Col%))
                ELSE
                    InputVarValue(z%) = SCENARIOMissingValueCode 'this is SCENARIO's Missing Value Code
                END IF
            ELSE
                IF (TRIM$(info$(FileNum%, Col%)) <> InputMissingValueCode$) AND
                    (LEN(TRIM$(info$(FileNum%, Col%))) > 0) THEN
                    InputVarValue(z%) = VAL(info$(FileNum%, Col%))
                ELSE 'missing value
                    IF FirstScene = TRUE THEN
                        InputVarValue(z%) = SCENARIOMissingValueCode
                    END IF
                END IF
            END IF
        END IF
    NEXT z%
END SUB

```

```

SUB GetNextVarName (varnames$, varname$, LastName)
    z% = INSTR(varnames$, ",")
    IF z% > 0 THEN
        varname$ = LEFT$(varnames$, z% - 1)
        varnames$ = RIGHT$(varnames$, LEN(varnames$) - z%)
    ELSE
        LastName = TRUE
        varname$ = varnames$
        varnames$ = ""
    END IF
    varname$ = LC$(TRIM$(varname$))
END SUB

SUB GetObservedData (FirstSim, FirstScene, info$(), InputMissingValueCode$) STATIC

    IF FirstScene = TRUE AND FirstSim = TRUE THEN
        NumObserved% = 6 'this is the number of recognized observed variables
        DIM observed(NumObserved%), ObservedName$(NumObserved%), ObservedUnit$(NumObserved%)
    END IF

    T = -1
    have.Tes = HAVE(FirstScene, have.Tes, IX.Tes)
    have.Tpill = HAVE(FirstScene, have.Tpill, IX.Tpill)
    have.Tre = HAVE(FirstScene, have.Tre, IX.Tre)
    have.Ttym = HAVE(FirstScene, have.Ttym, IX.Ttym)
    have.Tsk = HAVE(FirstScene, have.Tsk, IX.Tsk)
    have.HR = HAVE(FirstScene, have.HR, IX.HR)

    IF FirstScene = TRUE THEN
        IF have.Tes THEN ObservedName$(IX.Tes - NumInputVars%) = "Tes-obs": ObservedUnit$(IX.Tes -
NumInputVars%) = "(C)"
        IF have.Tpill THEN ObservedName$(IX.Tpill - NumInputVars%) = "Tpill-obs":
ObservedUnit$(IX.Tpill - NumInputVars%) = "(C)"
        IF have.Tre THEN ObservedName$(IX.Tre - NumInputVars%) = "Tre-obs": ObservedUnit$(IX.Tre -
NumInputVars%) = "(C)"
        IF have.Ttym THEN ObservedName$(IX.Ttym - NumInputVars%) = "Ttym-obs": ObservedUnit$(IX.Ttym -
NumInputVars%) = "(C)"
        IF have.Tsk THEN ObservedName$(IX.Tsk - NumInputVars%) = "Tsk-obs": ObservedUnit$(IX.Tsk -
NumInputVars%) = "(C)"
        IF have.HR THEN ObservedName$(IX.HR - NumInputVars%) = "HR-obs": ObservedUnit$(IX.HR -
NumInputVars%) = "(bpm)"
        NumObserved% = have.Tes + have.Tpill + have.Tre + have.Ttym + have.Tsk + have.HR
    END IF

    'get whatever data has been entered directly
    'note that missing data are carried over from the input file. This is
    'different from other input data in which previous values are carried down when
    'missing values are encountered.

    IF have.Tes THEN observed(z%) = ToDegC(IX.Tes)
    IF have.Tpill THEN observed(z%) = ToDegC(IX.Tpill)
    IF have.Tre THEN observed(z%) = ToDegC(IX.Tre)
    IF have.Ttym THEN observed(z%) = ToDegC(IX.Ttym)
    IF have.Tsk THEN observed(z%) = ToDegC(IX.Tsk)
    IF have.HR THEN observed(z%) = InputVarValue(IX.HR)
END SUB

SUB GetRecognizedVars
    'this routine is only run 1 time, the first time GetInputVarInfo is called

    'these are all global variables
    NumRecognizedVars% = 37
    DIM InputVarName$(NumRecognizedVars%)
    DIM InputVarFile(NumRecognizedVars%)
    DIM InputVarCol(NumRecognizedVars%)
    DIM InputVarUnit$(NumRecognizedVars%)
    DIM InputVarValue(NumRecognizedVars%)

    z% = 1
    InputVarName$(z%) = "ta": IX.Ta = z%: z% = z% + 1
    InputVarName$(z%) = "tmr": IX.Tmr = z%: z% = z% + 1
    InputVarName$(z%) = "tg": IX.Tg = z%: z% = z% + 1
    InputVarName$(z%) = "vair": IX.Vair = z%: z% = z% + 1
    InputVarName$(z%) = "rh": IX.RH = z%: z% = z% + 1

```

```

InputVarName$(z%) = "pvap": IX.Pvap = z%: z% = z% + 1
InputVarName$(z%) = "twb": IX.Twb = z%: z% = z% + 1
InputVarName$(z%) = "tdp": IX.Tdp = z%: z% = z% + 1
InputVarName$(z%) = "tnwb": IX.Tnwb = z%: z% = z% + 1
InputVarName$(z%) = "load": IX.load = z%: z% = z% + 1
InputVarName$(z%) = "grade": IX.grade = z%: z% = z% + 1
InputVarName$(z%) = "vmove": IX.Vmove = z%: z% = z% + 1
InputVarName$(z%) = "vo2": IX.VO2 = z%: z% = z% + 1
InputVarName$(z%) = "mtot": IX.Mtot = z%: z% = z% + 1
InputVarName$(z%) = "mrst": IX.Mrst = z%: z% = z% + 1
InputVarName$(z%) = "mext": IX.Mext = z%: z% = z% + 1
InputVarName$(z%) = "mwork": IX.MWork = z%: z% = z% + 1
InputVarName$(z%) = "pcteff": IX.PctEff = z%: z% = z% + 1
InputVarName$(z%) = "workmode": IX.WorkMode = z%: z% = z% + 1
InputVarName$(z%) = "bw": IX.BW = z%: z% = z% + 1
InputVarName$(z%) = "ht": IX.HT = z%: z% = z% + 1
InputVarName$(z%) = "pctfat": IX.PctFat = z%: z% = z% + 1
InputVarName$(z%) = "age": IX.Age = z%: z% = z% + 1
InputVarName$(z%) = "gender": IX.gender = z%: z% = z% + 1
InputVarName$(z%) = "phase": IX.Phase = z%: z% = z% + 1
InputVarName$(z%) = "accl": IX.Accl = z%: z% = z% + 1
InputVarName$(z%) = "vo2max": IX.VO2maxd = z%: z% = z% + 1
InputVarName$(z%) = "iclo": IX.Iclo = z%: z% = z% + 1
InputVarName$(z%) = "im": IX.Im = z%: z% = z% + 1
InputVarName$(z%) = "endmin": IX.endmin = z%: z% = z% + 1
InputVarName$(z%) = "rptmin": IX.rptmin = z%: z% = z% + 1

'----- observed variables - not used as model inputs -----
ix.FirstObserved% = z%
InputVarName$(z%) = "tpill": IX.Tpill = z%: z% = z% + 1
InputVarName$(z%) = "tre": IX.Tre = z%: z% = z% + 1
InputVarName$(z%) = "tes": IX.Tes = z%: z% = z% + 1
InputVarName$(z%) = "ttym": IX.Ttym = z%: z% = z% + 1
InputVarName$(z%) = "tsk": IX.Tsk = z%: z% = z% + 1
InputVarName$(z%) = "hr": IX.HR = z%: z% = z% + 1
END SUB

SUB GetUnits (varname$, unit$)
    z% = INSTR(varname$, "-")
    IF z% > 0 THEN
        unit$ = TRIM$(RIGHT$(varname$, LEN(varname$) - z%))
        varname$ = TRIM$(LEFT$(varname$, z% - 1))
    ELSE
        unit$ = ""
    END IF
END SUB

FUNCTION HAVE (FirstScene, HAVE.X, IX.X)
    IF FirstScene THEN
        HAVE.X = 0
    END IF

    IF HAVE.X = FALSE AND (InputVarFile(IX.X) > 0) THEN
        HAVE.X = InputVarValue(IX.X) <> SCENARIOMissingValueCode
    END IF

    HAVE = HAVE.X
END FUNCTION

FUNCTION JULIAN (Month, Day, Year)
    'check for leap year
    IF (Year MOD 4) = 0 THEN LeapYear = TRUE ELSE LeapYear = FALSE
    Date = Day - (Month > 1) * 31 - (Month > 2) * (LeapYear + 28) - (Month > 3) * 31
    Date = Date - (Month > 4) * 30 - (Month > 5) * 31 - (Month > 6) * 30
    Date = Date - (Month > 7) * 31 - (Month > 8) * 31 - (Month > 9) * 30
    Date = Date - (Month > 10) * 31 - (Month > 11) * 30
    JULIAN = Date
END FUNCTION

SUB PostInputData (Ta, Tmr, Pvap, Vair, Iclo, Im, BW, HT, SA, Mtot, Mext, Vmove, grade, load,
Age, StMin, PctFat, rptmin, endmin)
    CALL Post(4, 10, 2, 2, Ta)
    CALL Post(5, 10, 2, 1, Tmr)
    CALL Post(6, 10, 2, 1, Pvap)
    CALL Post(7, 10, 1, 0, Accl)

```

```

CALL Post(8, 10, 2, 1, Vair)
CALL Post(9, 11, 1, 2, Iclo)
CALL Post(10, 11, 1, 2, Im)
CALL Post(4, 23, 3, 1, BW)
CALL Post(5, 23, 3, 1, HT)
CALL Post(6, 22, 4, 1, Mtot)
CALL Post(7, 23, 3, 1, Mext)
CALL Post(8, 24, 2, 1, Vmove)
CALL Post(9, 24, 2, 1, grade)
CALL Post(10, 24, 2, 1, load)
CALL Post(4, 38, 3, 0, Age)
CALL ClrVal(5, 38): CALL Post(5, 38, 3, 1, StMin)
CALL Post(6, 39, 2, 1, PctFat)
CALL Post(7, 39, 2, 2, rptmin)
CALL Post(8, 38, 3, 1, endmin)
END SUB

SUB ReadAnotherRecord (NumInputFiles%, SubjectID$, nextinfo$(), info$(), NextDate(), CurrDate(),
EndDate, CurrentDate)
    FOR FileNum% = 2 TO NumInputFiles% + 1
        IF NextDate(FileNum%) > EndDate - dt / 24 / 60 / 2 AND NextDate(FileNum%) < EndDate + dt /
24 / 60 / 2 THEN
            CALL SetCurrentEqualNext(FileNum%, NumVarNames(FileNum%), nextinfo$(), info$(),
NextDate(), CurrDate())
            CALL ReadNextRecord(FileNum%, NumVarNames(FileNum%), SubjectID$, nextinfo$(),
NextDate())
            END IF

        NEXT FileNum%
        CurrentDate = EndDate
    END SUB

SUB ReadFirstRecords (NumInputFiles%, SubjectID$, info$(), CurrDate(), nextinfo$(), NextDate(),
CurrentDate)
    'read the first 2 records in the file
    FOR FileNum% = 2 TO NumInputFiles% + 1
        CALL ReadNextRecord(FileNum%, NumVarNames(FileNum%), SubjectID$, info$(), CurrDate())
        IF StartDate < CurrDate(FileNum%) THEN StartDate = CurrDate(FileNum%)
        CALL ReadNextRecord(FileNum%, NumVarNames(FileNum%), SubjectID$, nextinfo$(), NextDate())
        CurrentDate = StartDate
    NEXT FileNum%

    'keep reading records until the date of the current record is just less than
    'or equal to the startdate (i.e., nextdate is greater than startdate)
    FOR FileNum% = 2 TO NumInputFiles% + 1
        WHILE NextDate(FileNum%) <= StartDate AND EndFile(FileNum%) = FALSE
            CALL SetCurrentEqualNext(FileNum%, NumVarNames(FileNum%), nextinfo$(), info$(),
NextDate(), CurrDate())
            CALL ReadNextRecord(FileNum%, NumVarNames(FileNum%), SubjectID$, nextinfo$(),
NextDate())
        WEND
    NEXT FileNum%

    'Note: at this point, file pointers should be parked at the correct
    'record, info$ and date should correspond to this record, and nextinfo$
    'and NextDate should also be in hand
END SUB

SUB ReadNextRecord (FileNum%, NumVarNames, SubjectID$, info$(), CurrDate()) STATIC
    DIM LastInfo$(25)

    FOR z% = 1 TO NumVarNames
        LastInfo$(z%) = info$(FileNum%, z%)
    NEXT z%

    GoodRecord = FALSE
    WHILE GoodRecord = FALSE AND EndFile(FileNum%) = FALSE
        'read 1 record
        FOR z% = 1 TO NumVarNames
            IF NOT EOF(FileNum%) THEN
                INPUT #FileNum%, info$(FileNum%, z%)
            ELSE
                EndFile(FileNum%) = TRUE
            END IF
        NEXT z%
    WEND

```

```

'check the study ID
GoodStudy = TRUE
IF StudyIDCol(FileNum%) <> 0 THEN 'file contains study info
  IF Study$(FileNum%, info$()) <> StudyID$ THEN
    GoodStudy = FALSE
  END IF
END IF
'check the test IDs
GoodTest = TRUE
IF TestIDCol(FileNum%) <> 0 THEN 'file contains test info
  IF Test$(FileNum%, info$()) <> TestID$ THEN
    GoodTest = FALSE
  END IF
END IF
'check the subject IDs
GoodSubject = TRUE
IF SubjectIDCol(FileNum%) <> 0 THEN 'file contains subject info
  IF Subject$(FileNum%, info$()) <> SubjectID$ THEN
    GoodSubject = FALSE
  END IF
END IF
GoodRecord = GoodStudy * GoodTest * GoodSubject
WEND

'If no good records were found:
IF EndFile(FileNum%) = TRUE THEN
  FOR z% = 1 TO NumVarNames
    info$(FileNum%, z%) = LastInfo$(z%)
  NEXT z%
ELSE
  EndFile(FileNum%) = FALSE
  CurrDate(FileNum%) = Datee(FileNum%, info$())
END IF
END SUB

SUB ReadSimSpecFile (FirstSim, StudyID$, TestID$, SubjectID$, NumInputFiles%, InputFile$,
InputMissingValueCode$, info$(), DoneSession)
  STATIC NumDefaults, BeenHereBefore

  'Do this stuff the first time the procedure is called (i.e., when FirstSim=TRUE)
  IF BeenHereBefore = FALSE THEN
    BeenHereBefore = TRUE
    CALL GetFileName("simulation specification file name ", simspecfile$)
    simspecfile$ = "d:\obsdata\thermreg\test\simspec2.csv"
    OPEN simspecfile$ FOR INPUT AS #1
    'read any input variable names
    CALL GetInputVarInfo(1) '1 is the file number
  END IF 'end of stuff to do on FirstSim

  'read one record
  IF NOT EOF(1) THEN

    OutputFileName$ = ""
    INPUT #1, StudyID$, TestID$, SubjectID$, NumInputFiles%
    FOR z% = 1 TO NumInputFiles%
      INPUT #1, InputFile$(z%)
    NEXT z%
    INPUT #1, OutputFileName$, InputMissingValueCode$
    IF OutputFileName$ <> "" THEN FileOutputYN$ = "y" ELSE FileOutputYN$ = "n"
    OutputFileNum% = NumInputFiles% + 2

    'read default values
    FOR z% = 1 TO NumVarNames(1) 'NumVarNames(1) is the number of default var names
      INPUT #1, info$(1, z%)
    NEXT z%
  ELSE
    DoneSession = TRUE
  END IF
END SUB

SUB SetCurrentEqualNext (FileNum%, NumVarNames, nextinfo$(), info$(), NextDate(), CurrDate())
  FOR z = 1 TO NumVarNames
    info$(FileNum%, z) = nextinfo$(FileNum%, z)
  NEXT z
  CurrDate(FileNum%) = NextDate(FileNum%)
END SUB

```

```

FUNCTION Study$ (FileNum%, info$())
  IF StudyIDCol(FileNum%) <> 0 THEN Study$ = info$(FileNum%, StudyIDCol(FileNum%))
END FUNCTION

FUNCTION Subject$ (FileNum%, info$())
  IF SubjectIDCol(FileNum%) <> 0 THEN Subject$ = info$(FileNum%, SubjectIDCol(FileNum%))
END FUNCTION

FUNCTION Test$ (FileNum%, info$())
  IF TestIDCol(FileNum%) <> 0 THEN Test$ = info$(FileNum%, TestIDCol(FileNum%))
END FUNCTION

FUNCTION ToDegC (indexx)
  x = InputVarValue(indexx)
  IF x <> SCENARIOMissingValueCode THEN
    u$ = InputVarUnit$(indexx)
    IF u$ = "c" THEN u$ = ""
    SELECT CASE u$
      CASE IS = "f"
        ToDegC = 5 * (x - 32) / 9
      CASE IS = ""
        ToDegC = x
      CASE ELSE
        PRINT "bad units entered for " + InputVarName$(indexx)
        STOP
    END SELECT
  ELSE
    ToDegC = x 'if x=missing value code then do not convert
  END IF
END FUNCTION

FUNCTION ToMmHg (indexx)
  x = InputVarValue(indexx)
  IF x <> SCENARIOMissingValueCode THEN
    u$ = InputVarUnit$(indexx)
    IF u$ = "mmhg" THEN u$ = ""
    SELECT CASE u$
      CASE IS = "Pa"
        ToMmHg = x * 7.501 * 10 ^ (-4)
      CASE IS = "dynes/cm2"
        ToMmHg = x * 7.501 * 10 ^ (-3)
      CASE IS = "atm"
        ToMmHg = x * 760
      CASE IS = ""
        ToMmHg = x
      CASE ELSE
        PRINT "bad units entered for " + InputVarName$(indexx)
        STOP
    END SELECT
  ELSE
    ToMmHg = x 'if x=missing value code then do not convert
  END IF
END FUNCTION

FUNCTION ToMps (indexx)
  x = InputVarValue(indexx)
  IF x <> SCENARIOMissingValueCode THEN
    u$ = InputVarUnit$(indexx)
    IF u$ = "m/s" THEN u$ = ""
    'convert to m/s
    SELECT CASE u$
      CASE IS = "knots"
        ToMps = x * .5148
      CASE IS = "mph"
        ToMps = x * .44704
      CASE IS = "f/s"
        ToMps = x * .3048
      CASE IS = ""
        ToMps = x
      CASE ELSE
        PRINT "bad units entered for " + InputVarName$(indexx)
        STOP
    END SELECT
  ELSE
    ToMps = x 'if x is missing value, do not convert
  END IF
END FUNCTION

```

```

END IF

END FUNCTION

FUNCTION VarIndex% (varname$)
FOR z% = 1 TO NumRecognizedVars%
  IF varname$ = InputVarName$(z%) THEN
    VarIndex% = z%
    EXIT FOR
  END IF
NEXT z%
END FUNCTION

FUNCTION Watts (Index, x)
'convert x to Watts
SELECT CASE InputVarUnit$(Index)
  CASE IS = "w"
    Watts = x
  CASE IS = "w/m2"
    Watts = x * SA
  CASE IS = "met"
    Watts = x * Mrst
  CASE IS = "Kcal/h"
    Watts = x * 1.163056
  CASE IS = "cal/s"
    Watts = x * 4.187
  CASE IS = ""
    Watts = x
  CASE ELSE
    PRINT "bad units entered for " + InputVarName$(indexx)
    STOP
END SELECT
END FUNCTION

```